SIMECK 密码代数故障攻击研究 *

黄长阳 [1]，王　韬 [1]，陈　浩 [1]，王晓晗 [1]，马云飞 [1]，陈财森 [2]

(1. 陆军工程大学石家庄校区，石家庄 050003; 2. 陆军装甲兵学院，北京 100072)

摘　要：针对 SIMECK 密码给出一种代数故障攻击方法。首先给出 SIMECK 加密轮函数和密钥生成策略等效代数方程创建方法；分别设定故障已知模型和故障未知模型，并在故障未知模型下提出基于故障注入差分和基于正确/故障密文差分确定故障索引值两种策略创建故障信息方程；利用基于 SAT 问题求解方程组。结果表明，在 SIMECK32/64 第 24 轮注入单比特翻转故障，故障已知模型和基于故障注入差分的故障未知模型均仅需 2 次注入即可恢复完整 64 比特主密钥；在第 27 轮注入故障，基于密文差分的未知模型需 9 次注入可恢复完整主密钥。与已有研究相比，该攻击密钥搜索复杂度更低，所需故障注入样本量更少。

关键词：故障攻击；代数故障攻击；SIMECK 算法；轻量级分组密码；故障模型

中图分类号：TP309.2　　doi: 10.3969/j.issn.1001-3695.2018.01.0096

On algebraic fault attack against SIMECK cipher

Huang Changyang[1], Wang Tao[1], Chen Hao[1], Wang Xiaohan[1], Ma Yunfei[1], Chen Caisen[2]

(*Army Engineering University of PLA*, *Shijiazhuang* 050003, *China*; 2. *College of Armoured Force Engineering of PLA*, *Beijing* 100072, *China*)

Abstract: This paper evaluated the security of SIMECK using the algebraic fault analysis. Firstly, given the method of creating equivalent algebraic equations of SIMECK encryption round function and key generation strategy. Secondly, it designed the known fault model and stochastic fault model respectively, and proposed two strategies based on the differential value of fault injected and the fault index determined by differential value of ciphertext to create equivalent equations of fault information under the unknown model. Finally, solved equations based on the SAT problem. Experimental results show that after injecting single-bit fault to the 24th round of SIMECK32/64, the fault known model and the fault unknown model based on differential value of the fault injection, only 2 injections can recover the full 64-bit master keys. And 9 injections needed for the fault unknown model based on the differential value of ciphertext after injecting single-bit fault to the 27th round. The method's complexity of searching key is simpler and its fault injection sample required is less compared with the previous research.

Key words: fault attack; algebraic fault attack; SIMECK; lightweight block cipher; fault model

0　引言

故障攻击是旁路攻击的一种实现形式，以其高有效性受到研究者广泛关注。密码学家尝试将传统分析方法与故障攻击相结合，如差分攻击的提出者 Biham[1]将差分分析思想引入故障攻击，提出差分故障攻击，并成功破解 DES 密码。但差分故障攻击利用手工推导故障信息，无法将故障注入更深轮，因此故障信息利用率较低。

eSmart2010 会议上 Courtois 将其所提出的代数攻击思想引入故障攻击，提出代数故障攻击方法[2]并成功恢复 DES 密钥。代数故障攻击利用经典代数分析与故障攻击各自的优势，弥补两种方法单一使用时的不足，即利用代数分析思想将故障信息表示成代数方程组，能够充分利用可计算资源以提高故障信息利用率；同时所创建的故障信息方程能够有效降低代数分析中方程组的求解复杂度。之后研究人员将代数故障攻击成功扩展至 PRESENT[3]、GOST[4]、LED[5]等密码算法，进一步证明了代数故障攻击的有效性。

本文攻击对象 SIMECK 密码[6]是一种专为运行在硬件资源有限的无源 RFID 标签等设计的轻量级分组密码，采用典型 Feistel 结构，加密轮函数内部舍弃设计较为成熟的 S 盒，而是将按位“与”运算、“异或”运算和循环移位操作组合而成，具有优秀的硬件实现效率，更加符合分组密码轻量化设计趋势[7]。

收稿日期：2018-01-12；**修回日期：**2018-03-22　　**基金项目：**国家自然科学基金资助项目（61272491，61309021，61402528）

作者简介：黄长阳（1994-），男，黑龙江望奎人，硕士研究生，主要研究方向为轻量级分组密码代数故障攻击（13089998121@163.com）；王韬（1964-），男，教授，博导，博士，主要研究方向为信息安全、密码学；陈浩（1987-），男，博士研究生，主要研究方向为对称密码旁路分析；王晓晗（1992-），男，博士研究生，主要研究方向为信息安全与对抗；马云飞（1992-），男，硕士研究生，主要研究方向为密码学立方攻击；陈财森（1983-），男，讲师，博士，主要研究方向为信息安全与对抗．

SIMECK32/64 密码硬件实现仅需 549GE，小于轻量级分组密码 ISO 标准算法 PRESENT 的 1570GE。目前针对 SIMECK 密码的攻击主要集中在线性分析[8] [9]和差分分析[10] [11]等传统密码分析方法，最好结果将密钥搜索空间降至 2^{29}，很难对密码构成实际威胁。故障攻击方面，Saraswat 等利用差分故障攻击[12]，通过在倒数第 2 轮注入单比特故障，平均 28.32 个故障能够恢复最后一轮 16 位轮密钥。但该方法存在以下不足：a）仅能恢复末轮 16 位密钥，但根据 SIMECK 密码的密钥扩展算法特性，需获得连续 4 轮轮密钥才可能恢复全部主密钥信息；b）需要手工计算故障传播路径，不能充分利用可计算资源，故障注入深度较浅使得可利用故障加密信息较少，因此需要注入故障数量较多，导致故障信息利用率较低。

本文提出针对 SIMECK32/64 密码的代数故障攻击，在密码右寄存器注入单比特翻转故障：a）假设故障已知模型，在倒数第 9 轮注入单比特故障，仅需 2 次故障注入即可恢复 64bit 完整主密钥信息；b）假设故障未知模型，分别采用两种策略创建故障信息方程:(a)创建故障注入差分方程，结果需 2 次故障即可恢复 64bit 完整主密钥信息；(b)通过匹配密文差分表确定故障索引值，结果需要 9 次故障即能恢复 64 比特主密钥。

1 相关知识

1.1 SIMECK 密码介绍

SIMECK 算法是 CHES2015 上提出的一种轻量级分组密码，不同版本根据分组长度和密钥长度的差异表示为 SIMECK $2n/4n$（n 为加密字长度，n=16,24,32），其中 $2n$ 为分组长度，$4n$ 为密钥长度，对应加密轮数分别为 32、36 和 44。以 SIMECK 32/64 为例，表示该版本分组长度为 32bit，密钥长度为 64bit，加密迭代 32 轮。并且全部版本均满足无源 RFID 标签对硬件实现面积、吞吐量及功耗的要求[13]。

本文所使用符号含义说明如表 1 所示。

表 1 文中符号代表含义

符号	含义
$X<<a$	比特序列 X 循环左移 a 位
&	按位“与”运算
⊕	按位“异或”运算
mk_i	主密钥第 i 比特，$0\leq i<4n$
mL_i	明文左寄存器第 i 比特，$0\leq i<n$
mR_i	明文右寄存器第 i 比特，$0\leq i<n$
$sk_r[i]$	第 r 轮子密钥第 i 比特，$0\leq i<n$
$xL_r[i]$	加密至第 r 轮，左寄存器状态第 i 比特，$0\leq i<n$
$xR_r[i]$	加密至第 r 轮，右寄存器状态第 i 比特，$0\leq i<n$
$xL^*_r[i]$	注入故障后，第 r 轮左寄存器状态第 i 比特，$0\leq i<n$
$xR^*_r[i]$	注入故障后，第 r 轮右寄存器状态第 i 比特，$0\leq i<n$

1.1.1 加密轮函数设计

SIMECK 算法加密整体采用典型 Feistel 结构，轮函数内加密部件由“与”运算、“异或”运算和循环移位操作组成，如图 1 所示。其中“与”运算能够增强算法非线性，并和“异或”运算共同作用使得加密运算在两寄存器间扩散，循环移位操作使得加密运算在寄存器内部扩散。

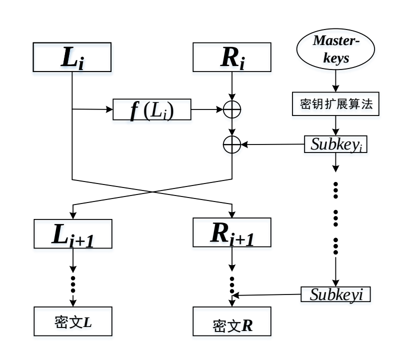


图 1 SIMECK 族密码加密过程

以 SIMECK32/64 版本为例，具体加密过程如下：

a) 将 32bit 明文均分为左右两部分，记为 mL 和 mR。分别将 mL 和 mR 作为 SIMECK 密码左寄存器和右寄存器的初始状态；

b) 启动加密算法，SIMECK 加密轮函数对寄存器内明文进行迭代运算，32 轮加密后寄存器内部状态即为加密密文；密码寄存器状态更新如式(1)所示。

$$\left.\begin{aligned} xR_{r+1} &= xL_r \\ xL_{r+1} &= f\left(xL_r\right)\oplus R_r \oplus sk_r \end{aligned}\right\} \tag{1}$$

其中:f 函数运算表达式如式(2)所示。

$$f\left(x\right)=\left(x\,\&\,(x<<5)\right)\oplus(x<<1) \tag{2}$$

1.1.2 密钥生成策略

SIMECK 密钥扩展算法借鉴 NSA 提出的 SPECK 密码的设计思路，重用轮函数计算轮子密钥，这样设计的优势是通过使用轮函数的硬件实现来进行密钥扩展计算，无需另外设置硬件实现密钥扩展算法，提高安全性的同时减少硬件资源消耗，更符合轻量级密码的设计准则，以更好的运行于资源受限设备。此外还引入由 LFSR 生成的 m-序列编排轮子密钥以增强密码安全性。

以 SIMECK32/64 版本为例，轮子密钥生成过程如下：

a) 将 64 bit 主密钥均分成 4 部分，记为（t_2，t_1，t_0，k_0），其中 k_0 为主密钥的最高 16 位，t_2 为主密钥最高 16 位。分别作为密钥扩展寄存器 $T2$，$T1$，$T0$，$K0$ 的初始状态；

b) 将寄存器 T0 和 K0 的内部状态送入加密轮函数，并引入常量 C 和 m-序列 Z_j，运算结果作为 $T2$ 寄存器下一轮的更新状态，如式（3）所示。

$$\left.\begin{aligned} K0_{r+1} &= T0_r \\ T2_{r+1} &= K0_r \oplus f\left(T0_r\right)\oplus C\oplus (z_j)_r \end{aligned}\right\} \tag{3}$$

每轮运算后，密钥反馈移位寄存器右移 n 位。其中，r 为加密轮数，C 为加密常量，$C=2^n-4$，$(Z_j)_r$ 为 m-序列 Z_j 的第 r 比特，各版本密码所用 Z_j 序列的初始状态及反馈特征多项式详见文献[12]。每轮扩展运算后，将寄存器 $K0$ 的内部状态作为本轮加密轮密钥。

1.2　代数故障攻击原理

针对密码算法进行代数故障攻击关键思想可总结成三部分：首先，将密码正确加密过程表示成代数方程组；重启加密算法，诱导中间轮发生故障并将故障信息转换成等效方程组；最后联立方程组进行求解。攻击框架如图 2 所示。

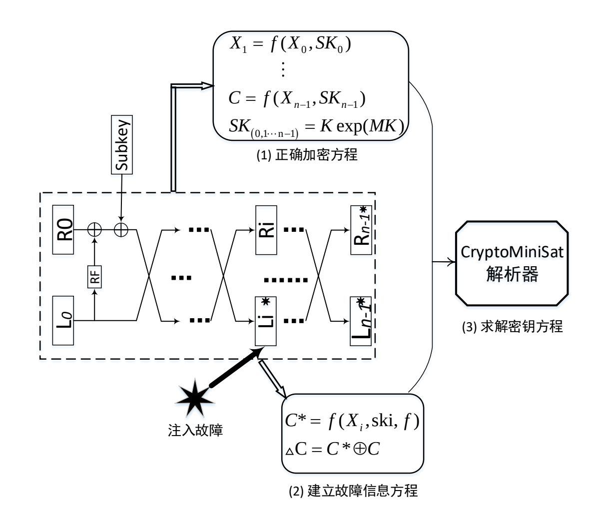


图 2　代数故障攻击原理

代数故障攻击首先运用典型代数攻击思想，分析密码加密轮函数及密钥扩展算法数学结构以及其中加密部件固有数学特征，创建关于明/密文、主密钥、轮密钥以及加密轮中间状态的等效代数方程组。

重启加密算法，对密码设备注入故障即通过更改加密设备瞬时电压或聚焦光脉冲照射密码芯片等方式致使寄存器内部状态发生改变[14]。分析所注故障随加密扩散过程中与其应用轮密钥之间相关性，将故障信息表示为代数方程并与密码正确加密方程联立，以降低方程组求解复杂度。利用故障信息的前提是设定有效的故障模型，故障模型是对所注入故障的抽象描述，关键参数包含故障作用效果、故障深度即故障注入轮至故障密文输出所经加密轮数和故障宽度即一次故障注入能影响所注轮比特位数等[15]。

最后联立方程组，并运用基于 Grŏbne 基、线性化及其扩展算法、混合整数规划问题、伪随机优化问题、基于可满足性(Satisfiability，SAT)问题等方法求解方程。其中，基于 SAT 问题方法适用于任何次数的多元方程，更适合密码方程求解，因此本文选取该方法。

2　SIMECK32/64 等效方程创建

首先，正确运行 SIMECK 加密算法，将加密过程转换成等效代数方程，关键是如何构建关于加密轮函数和轮密钥生成策略的代数方程组。

2.1　轮函数方程

分析 SIMECK 密码轮函数结构及其内部加密组件数学特性，创建方程如式(4)所示。

$$\left.\begin{aligned}&xR_{r+1}[i]\oplus xL_r[i]=0;\\&xL_{r+1}[i]\oplus\big(xL_r[i]\,\&\,(xL_r[i]<<5)\big)\oplus\big(xL_r[i]<<1\big)\oplus xR_r[i]\oplus sk_r[i]=0\end{aligned}\right\}\quad(4)$$

为简化方程表示以提高求解效率，引入 3 组 16 bit 中间变量 X_1，X_2，X_3 辅助方程表示，如式（5）所示。

$$\left.\begin{aligned}&xL_r[i]<<1=X_1[i];\\&xL_r[i]<<5=X_2[i];\\&xL_r[i]\,\&\,X_2[i]=X_3[i];\end{aligned}\right\}\quad(5)$$

其中，$0\le r<32$，$0\le i<16$。加密总迭代 32 轮，因此共创建轮函数等效方程 5×16×32=2560 个，引用变量共(32+16+3×16) ×32=3072 个。

2.2　密钥扩展等效方程

由于SIMECK密码轮密钥生成策略是通过重用轮函数并引入轮常量与 m-序列对主密钥进行运算，因此密钥寄存器 $T2$ 内部状态更新等效方程创建方法与轮函数加密类似，其余寄存器内状态通过寄存器移位进行更新，因此创建密钥寄存器状态更新等效方程如式（6）所示。

$$\left.\begin{aligned}&K_{r+1}[i]\oplus T0_r[i]=0;\\&T0_{r+1}[i]\oplus T1_r[i]=0;\\&T1_{r+1}[i]\oplus T2_r[i]=0;\\&T2_{r+1}[i]\oplus K_r[i]\oplus(T0_r[i]\,\&\,T0_r[i]<<5)\oplus(T0_r[i]<<1)\oplus C\oplus(z_j)_r=0\end{aligned}\right\}(6)$$

其中 K，$T0$，$T1$，$T2$ 分别代表密钥生成反馈移位寄存器的四个寄存器内部状态。与创建轮函数方程相似，引入 3 组新变量简化方程表示，如式（7）所示。

$$\left.\begin{aligned}&T0_r[i]<<1=Y_1[i];\\&T0_r[i]<<5=Y_2[i];\\&T0_r[i]\,\&\,Y_2[i]=Y_3[i];\end{aligned}\right\}\quad(7)$$

需特别说明的是，由于 SIMECK 算法前四轮加密的轮密钥使用 64 比特主密钥，所以只需运行 28 轮密钥扩展算法即能够获得全轮加密所需完整轮密钥。因此共创建等效方程 7×16×28=3136 个，所引用变量共(4×16+3×16)×28+16+28=3180 个。

3　故障信息分析

求解经典代数攻击所创建方程的复杂度会随着加密迭代轮数的增多而呈指数增长，而轻量级分组密码往往通过增多加密轮数来提高安全性，求解密码全轮方程被证明是 NP 完全问题，现有计算能力无法在指数时间内完成求解。分析故障信息的目的正是通过创建故障信息方程以降低密钥方程求解复杂度。

3.1　故障注入寄存器选取

分析 SIMECK 密码加密整体采用平衡 Feistel 结构及轮函

数的运算特性，每轮加密时仅将左寄存器内部状态送入 f 函数进行“与”、“异或”和循环移位操作的混合运算，并将运算结果作为下一轮左寄存器的更新状态。而右寄存器内部状态则不经过任何加密操作，直接等于上一轮左寄存器内部状态。左寄存器和右寄存器内部 1 比特故障影响下一轮状态情况分别如图 3、4 所示。

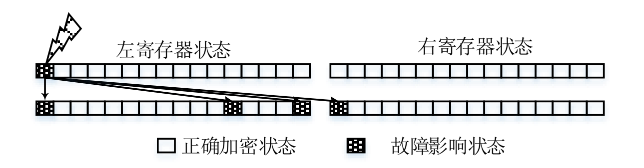


图 3　左寄存器内故障对下轮影响情况

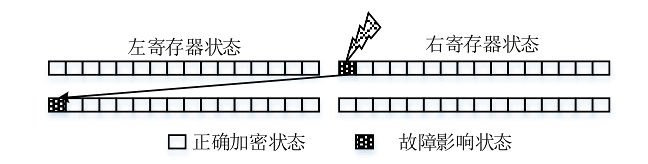


图 4　右寄存器内故障对下轮影响情况

观察图 3、4 可得出，在左寄存器注入故障时，故障能够影响下一轮中间状态更多比特位，即故障能够更快扩散至下一轮。但同时，通过对比发现在加密第 r 轮左寄存器第 i 比特注入故障等价于加密第 r-1 轮右寄存器第 i 比特注入故障，因此同等条件下右寄存器故障注入可比左寄存器更深一轮，即能够利用更深一轮的故障信息，因此本文选取密码右寄存器进行故障注入

3.2　故障已知模型

本文采用单比特翻转故障，首先设定故障已知模型，即准确控制故障诱导比特位。通过分析 SIMECK 密码使用“&”、“异或”及循环移位操作混合运算使得加密扩散原理，推导从故障注入加密轮至故障密文输出过程中故障扩散路径，以向右寄存器第 8 bit 注入单比特故障为例，所注故障随迭代加密扩散效果如图 5 所示。

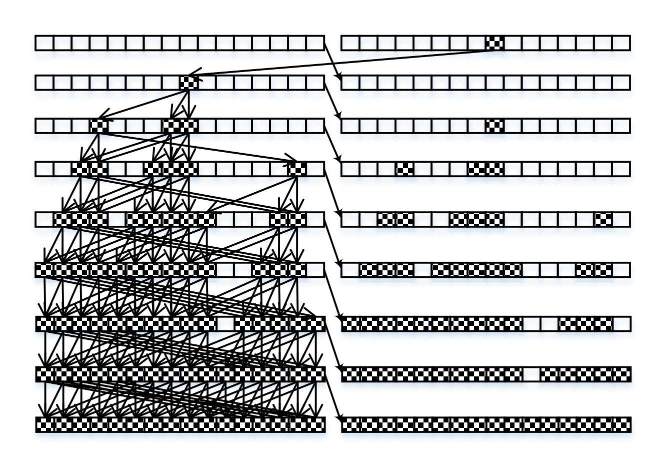


图 5　SIMECK32/64 单比特故障扩散路径

观察图 5 可得出，在密码右寄存器注入单比特故障时，经过 7 轮迭代运算后故障能够扩散至左寄存器状态全部 16 比特，再经过一轮加密即可扩散至全部 32 比特密文。因此，得到正确密文输出后，在同一明文和主密钥条件下重启加密算法，加密运算至第 24 轮时向右寄存器中间状态注入单比特故障。

获得故障密文后，按照第 2 节方法将故障注入轮至故障密文输出加密过程转化成等效代数方程组。同时分析故障扩散过程中所使用轮密钥与故障中间状态的相关性，创建图 5 中故障扩散过程等效代数方程。以第 24～25 轮加密为例，可构建方程如式（8）所示。

$$\left.\begin{aligned}
&xL^*_{24}[i]\oplus xL_{24}[i]=0, 0\le i<16;\\
&xR^*_{24}[i]\oplus xR_{24}[i]=0, 0\le i<16, i\ne 8;\\
&xL^*_{25}[8]\oplus xL_{24}[8]\,\&\,xL_{24}[13]\oplus xL_{25}[9]\oplus xR^*_{25}[8]\oplus sk_{24}[8]=0;\\
&xL^*_{25}[i]\oplus xL_{25}[i]=0, 0\le i<16, i\ne 8;\\
&xR^*_{25}[i]\oplus xR_{25}[i]=0, 0\le i<16;\\
&xL^*_{26}[3]\oplus xL_{25}[3]\,\&\,xL^*_{25}[8]\oplus xL_{25}[4]\oplus xR_{25}[3]\oplus sk_{25}[3]=0;\\
&xL^*_{26}[7]\oplus xL_{25}[7]\,\&\,xL_{25}[12]\oplus xL^*_{25}[8]\oplus xR_{25}[7]\oplus sk_{25}[7]=0;\\
&xL^*_{26}[8]\oplus xL^*_{25}[8]\,\&\,xL_{25}[13]\oplus xL_{25}[9]\oplus xR^*_{25}[8]\oplus sk_{25}[8]=0;
\end{aligned}\right\}\quad(8)$$

剩余轮故障传播信息方程构建原理与上式相似，在此不一一赘述。

3.3　故障未知模型

在故障已知模型有效基础上，本文提出故障未知模型，即无需精确控制故障具体发生在寄存器哪一比特位，因此不能通过建立如式(8)方程利用故障信息。本文提出两种策略利用未知模型下的故障信息。

3.3.1 创建故障注入差分方程

差分故障攻击若将故障注入较深轮时，受手工推导能力限制难以预测故障注入位置，导致故障攻击失效。本文给出一种通过创建故障注入差分值代数方程表示所注故障的方法。

选取加密第 24 轮右寄存器随机注入单比特故障，将正确加密时右寄存器内部状态记为 $xR_0|xR_1|\dots|xR_{15}$，诱导故障后内部状态记为 $xR^*_0|xR^*_1|\dots|xR^*_{15}$。引入 16 比特新变量 $\Delta x_i(0\le i<16)$ 表示故障注入轮正确/故障状态差分值，如式（9）所示。

$$\Delta x_i = xR_i \oplus xR^*_i \quad (9)$$

如果故障注入在第 i 比特，则 $\Delta x_i = xR_i \oplus xR^*_i = 0$。由于每次攻击只注入 1 比特故障，所以 Δx_0～Δx_{15} 中有且仅有 1 个值为 1，其余 15 比特值均为 0，基于此，引入 16 比特中间变量 $tmp_i(0\le i<16)$，令 $tmp_i = 1\oplus\Delta x_i$。则 tmp_0～tmp_{15} 中有且只有 1 比特值为 0，其余 15 比特值均为 1。因此可将故障注入信息表示为关于变量 tmp 的方程，如式(10)所示。

$$\left.\begin{aligned}
&tmp_0\vee tmp_1\vee\cdots\vee tmp_{15}=1;\\
&tmp_m\vee tmp_n=1, 0\le m<n<16;
\end{aligned}\right\}\quad(10)$$

至于左寄存器内部状态，并没有发生故障，因此可创建方

程如式（11）所示。

$$xL_i \oplus xL^*_i = 0\ ,0 \le i < 16 \tag{11}$$

因此，通过引入 16 比特中间变量 *tmp*，共创建 137 个等式将故障注入差分表示成代数方程组。

3.3.2 根据正确/故障密文差分确定故障索引值

上节方法中虽能灵活运用代数方程表示故障注入轮差分，但是无法确定具体故障注入位置，就无法利用故障扩散信息。本文通过分析故障传播特性，提出一种根据正确/故障密文差分值确定故障索引值的方法。

在加密倒数第 6 轮右寄存器注入单比特故障，当故障注入比特位不同时故障扩散路径不同，输出故障密文与正确密文差分值中 0 比特和 1 比特的位置均不相同，基于此原理本文提出通过匹配正确/故障密文差分值确定故障索引值方法。按照 3.2 节方法，依次分析推导故障注入位置从第 0 比特至第 15 比特，计算正确/故障密文差分值，结果如表 2 所示。

表 2　正确/故障密文差分对照表

故障位置	正确/故障密文差分值
R_0	1,1,0,0,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1,0,1,1,1
R_1	1,1,1,0,0,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1,0,1,1
R_2	1,1,1,1,0,0,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1,0,1
R_3	1,1,1,1,1,0,0,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1,0
R_4	1,1,1,1,1,1,0,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1
R_5	1,1,1,1,1,1,1,0,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1,0,0,0,1,1,0,0,1,1
R_6	1,1,1,1,1,1,1,1,0,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1,0,0,0,1,1,0,0,1
R_7	1,1,1,1,1,1,1,1,1,0,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1,0,0,0,1,1,0,0
R_8	1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,0,0,1,1,1,0,1,1,1,1,1,0,0,0,1,1,0
R_9	0,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,0,0,1,1,1,0,1,1,1,1,1,0,0,0,1,1
R_{10}	1,0,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,0,0,1,1,1,0,1,1,1,1,1,0,0,0,1
R_{11}	1,1,0,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,0,0,1,1,1,0,1,1,1,1,1,0,0,0
R_{12}	1,1,1,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1,0,1,1,1,1,1,0,0
R_{13}	0,1,1,1,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1,0,1,1,1,1,1,0
R_{14}	0,0,1,1,1,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1,0,1,1,1,1,1
R_{15}	1,0,0,1,1,1,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1,0,1,1,1,1

由表 2 可得出，当故障注入在 R_0～R_{15} 不同位置时，正确/故障密文差分值均不同。因此获取故障密文后，将密文差分值与表 2 内容逐一比较，若与第 *i* 行差分值匹配相同，则可推断出故障注入在 R_i 比特进而利用故障扩散信息。

但实验过程中注入故障后所输出密文与正确密文差分值与理论推导值并不完全相同：以明文=0xf029 0216，主密钥=0x1918 1110 0908 0100 进行加密，正确密文为 0xa4b1 bab9；在第 R0 注入单比特故障后输出故障密文=0x2eef 6acd，密文差分值=0x8a5e 0074，转换成二进制为 1000 1010 0101 1110 0000 0000 0111 0100，与表 2 中第 R_0 行差分值并不完全相同。分析原因，是由于 SIMECK 轮函数内“&”运算的数学特性会导致某些情况下故障传播失效，具体可分为以下三种情况：

假设“&”加密运算双方为 X_1 和 X_2，诱导故障后状态记为 X^*_1 和 X^*_2：

a）X_1=1，X_2=1，正确加密结果 X_1&X_2=1。无论故障注入是 X_1 还是 X_2，原状态由 1 跳变为 0，故障加密结果 X_1^*&X_2=0，状态翻转，故障成功扩散至下一比特，如图 6 所示。

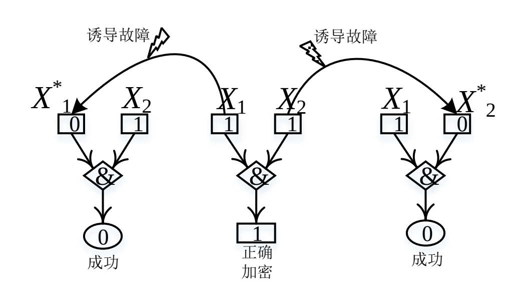


图 6　“&”运算故障扩散成功情况

b）X_1=1，X_2=0，正确加密结果 X_1&X_2=0。当故障注入在 X_2 时，故障加密结果 X_1&X_2^*=1，状态翻转，故障传播成功；但是当故障注入在 X_1 时，故障加密结果 X_1^*&X_2=0，与正确加密相同，故障传播失效，如图 7 所示。

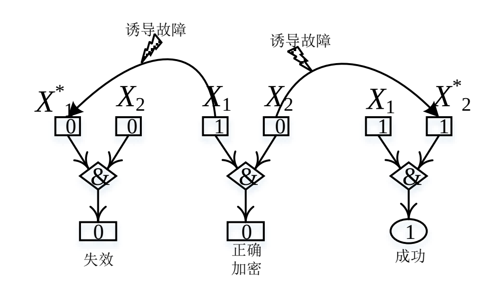


图 7　“&”运算故障失效情况 1

c）X_1=0，X_2=0，正确加密结果 X_1 &X_2=0。此时，不管故障注入在 X_1 还是 X_2，加密结果 X_1*&X_2= X_1 &X_2*=0，均与正确加密结果相同，故障传播失效，如图 8 所示。

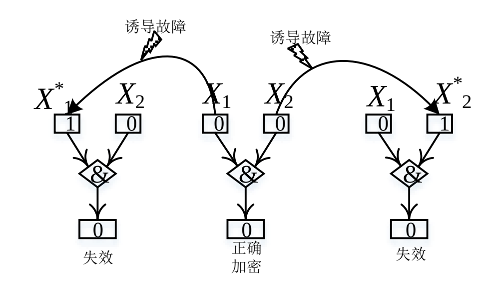


图 8　“&”运算故障失效情况 2

鉴于上述情况，本文提出以下策略确定故障索引：

a）设置 16 个计数器，记为 *Counter*[0]～*Counter*[15]，分别对应表 2 中第 R_0～R_{15} 行，并将计数器初始值置零；

b) 获得正确密文和故障密文输出后，计算密文差分值。将正确/故障密文差分依次与表 2 第 R_0～R_{15} 行内容逐比特进行比较，当第 R_i 行每匹配成功 1 比特则所对应计数器 *Counter*[*i*] 值加 1，最终得到 16 个 *Counter*[*i*] 的值；

c) 比较 *Counter*[*i*] (0≤i＜16) 值大小，选取其中最大值 *Counter_max*[*i*] 所对应表 2 第 R_i 行为最终成功匹配成功行，即确定故障注入位置在第 R_i 比特。算法如表 3 所示。

表 3　基于密文差分确定故障索引值算法

```
输入：正确密文 C 和故障密文 C*
输出：故障诱导位置 location
--------------------------------
n←16;
Counter[i] ←0，0≤i<16;
1. 随机生成明文 P 和主密钥 MK，正确加密并输出密文 C;
2. 在同一 P 与 MK 条件下，重启加密算法，向第 27 轮右寄
   存器注入单比特故障，加密至故障密文 C*;
for i=0 to i=2n do
    ΔX[i]=C[i]⊕C*[i];
for r=0 to r=n do
    for i=0 to i=2n do
        if ΔX[i]=R_r[i]
            then   Counter[r]++;
    end for
end for
Counter_max[r] = max(Counter[0]～Counter[15]);
loationc = r;
return   location
```

为验证上述所提策略有效性，进行 1000 次攻击实验，每次实验分别依次对第 R_0～R_{15} 比特诱导故障，按照表 3 算法确定故障注入位置，其中能够正确推导故障注入位置共 13259 次，成功率 $\rho = 13259/(16 \times 1000) \approx 82.9\%$。当故障索引值确定后，即能够利用式（7）创建故障扩散方程。

4　攻击实验

在普通 PC 机（CPU：Intel(R) Core™ i5-4710 @3.20 GHz，内存 4 GB，64 位 Windows 7 系统）上，使用 C++语言编程仿真故障注入过程，使用 CryptoMinisat 解析器（4.4 版本）求解密钥方程。攻击实验流程如图 9 所示。

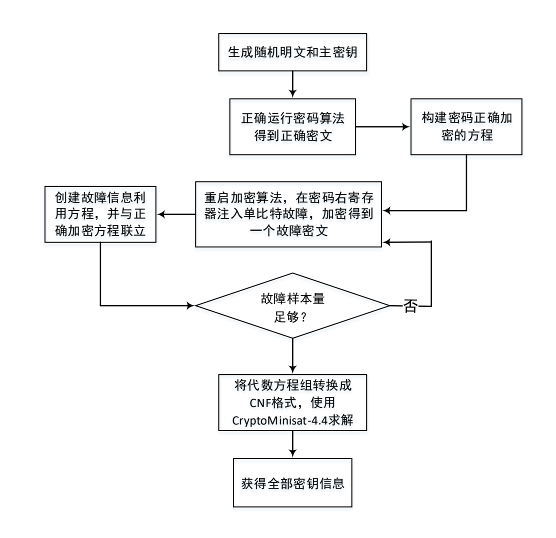


图 9　SIMECK 密码攻击实验过程

4.1　SIMECK32/64 攻击实例

a) 开始生成 32bit 明文 *mP*=0x eb63 8896，随机 64bit 主密钥 *mk*=0x ee8b 0598 6b06 6cc3，正常启动加密算法，得到正确密文 *C*=0xea6f 0387，使用第 2 节方法创建正确加密等效代数方程组；

b) 同一明文和密钥条件下重启加密算法，故障已知模型设定下，加密至第 24 轮时向右寄存器注入单比特故障，加密至故障密文输出，重复该步骤直至获得最后样本量。两种故障未知模型设定下，分别在加密第 24 轮和第 27 轮右寄存器注入单比特故障。按照第 3 节方法建立故障信息方程。

c) 使用 CryptoMinisat 解析器求解方程，结果如图 10 所示。64 bit 主密钥编号为 46～109，其中正数表示该编号所代表变量值为 1，负数表示该编号变量所代表的变量值为 0。因此求解出的密钥信息为：1110 1110 1000 1011 0000 0101 1001 1000 0110 1011 0000 0110 0110 1100 1100 0011，转换为十六进制为 0xee8b 0598 6b06 6cc3，与正确密钥信息一致，攻击成功。

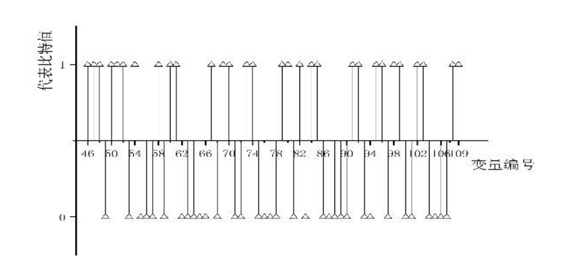


图 10　CryptoMinisat 求解结果

4.2　实验结果分析

为充分验证所提方法有效性，故障已知模型设定下，增加故障注入数量，分别进行 100 次攻击实验，平均求解时间如图 11 所示。

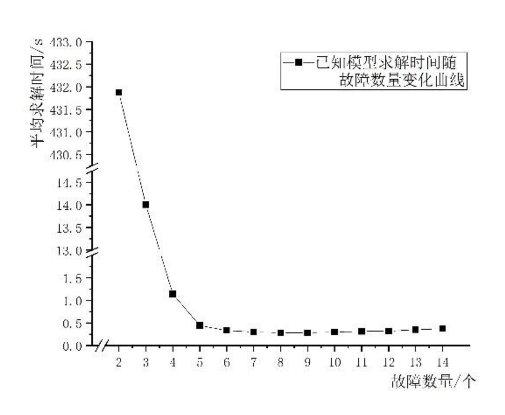


图 11　已知模型设定下平均求解时间

a) 由图 11 可知，2 次故障注入即可恢复完整 64 bit 主密钥，平均需 431 s，3 次故障注入可将求解时间下降至 14 s；

b）前期，随着故障注入数量的增加，平均求解时间逐渐减小，且故障样本量越少时间减小趋势越明显，这是由于注入故障数量增加使得可创建故障信息方程增加，从而充分降低方程求解复杂度；当故障数量 n=9 时求解时间将至最少，仅为 0.28s，当 $n\geq9$ 时，故障数量增加时，求解时间反而略有增长，这是因为故障样本量过多时，导致产生的故障信息方程冗余，使得求解时间变长。

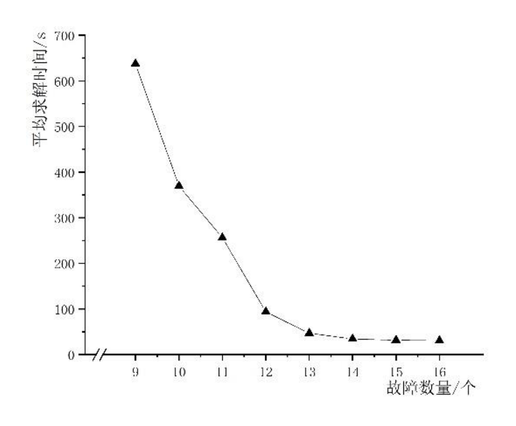


图 12　基于密文差分的未知模型平均求解时间

基于正确/故障密文差分的未知模型下，最少需 9 次注入可恢复全部 64 bit 主密钥，平均需 637 s。不同故障样本量下平均求解时间如图 12 所示。

对比图 11 和 12 的曲线，发现两种模型下求解时间随故障样本量变化趋势大致相同，这是因为两种模型利用故障信息原理相似，均能够创建故障传播信息方程；但相同故障样本量下，未知模型平均求解时间均大于已知模型，因为为了能够确定故障索引值，未知模型故障注入深度为 6，小于已知模型深度 9，因此每次故障注入后未知模型可利用的故障加密方程和故障传播方程均小于已知模型，致使求解复杂度较高。

基于故障注入差分的未知模型下，仅需 2 次故障注入即可恢复完整主密钥，100 次攻击实验求解时间分布如图 13 所示。

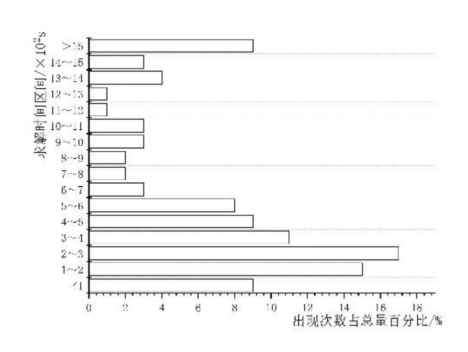


图 13　基于故障注入差分的未知模型求解时间分布（n=2）

由图 13 可得，79%的实验能够在 1000s 内完成求解，总平均求解时间为 681s，大于已知模型平均求解时间。虽然两种故障模型设定故障深度相同，所能够利用故障加密过程等效方程相同，但是该未知模型仅利用故障注入差分信息，相比已知模型能够利用故障传播信息所能利用故障信息较少，使得方程求解复杂度略高。

4.3　与已有研究比较

本文攻击成果与已有故障攻击研究比较如表 4 所示。

表 4　SIMNECK32/64 故障攻击结果比较

研究来源	攻击方法	故障数量	恢复密钥属性
文献[12]	单比特随机故障，深度 f_d=2，根据故障密文推导末轮轮密钥	28.32	16 比特末轮轮密钥
本文	单比特故障已知，深度 f_d=9，创建故障加密及故障传播方程	2	完整 64 比特主密钥
本文	单比特随机故障，深度 f_d=9，基于故障注入差分创建故障信息方程	2	完整 64 比特主密钥
本文	单比特随机故障，深度 f_d=6，基于密文差分确定故障索引值	9	完整 64 比特主密钥

对比文献[12]，本文代数故障攻击结果优点如下：

a) 已知模型和两种未知模型设定下，均能恢复完整 64 比特主密钥信息；文献[12]中利用差分故障攻击，成功恢复最后一轮 16 比特轮密钥。但根据 SIMECK 密钥生成策略，需要获得连续 4 轮轮密钥才可能恢复全部主密钥。本文通过创建密钥扩展算法等效方程，三种故障模型下均能成功恢复完整主密钥信息，密钥搜索复杂度更低，对密码安全性威胁更大；

b) 故障注入轮更深，故障信息利用率更高，所需故障注入数量更少；文献[12]差分故障攻击中，在倒数第 2 轮注故障，每次故障注入可恢复末轮密钥的 2 比特，平均需要 28.32 个故障能够恢末轮 16 比特密钥。由于 SIMECK 密码轮函数中按位“与”、“异或”和循环移位操作的混合运算使得故障传播特征

较为复杂，当故障注入过深轮时，差分故障攻击利用繁琐手工推导难以确定故障位置，使得故障攻击失效。而本文能够灵活的将故障信息表示成代数方程组形式，两种策略下的故障未知模型分别通过创建故障注入差分方程表示故障注入信息和通过匹配正确/故障密文差分确定故障索引值，充分利用可计算资源将故障注入更深轮，三种模型下分别将故障深度 f_d 加深至 9,9 和 6，使得可利用故障信息增多，分别仅需 2 次、2 次和 9 次故障即可恢复完整 64 比特主密钥，攻击复杂度更低。

5 结束语

本文针对 SIMECK32/64 密码进行代数故障攻击，设定三种故障模型，分别需要 2 次、2 次和 9 次故障注入即能恢复完整 64 bit 主密钥信息，最佳求解时间仅为 0.32 s。这表明在数学上设计安全的 SIMECK 密码并不能有效抵御故障攻击的威胁，同时为后续完善密码安全性设计提供有益参考价值和借鉴意义。由于 SIMECK 算法加密轮函数中“&”运算的数学特性可能会使得故障扩散过程中出现失效，因此今后将深入分析故障失效情况以进一步提高故障信息利用率。

参考文献：

[1] Biham E, Shamir A. Differential fault analysis of secret key cryptosystems [C]// Proc of Crypto. 1997: 513-525.

[2] Courtois N, Ware D, Jackson K. Fault-algebraic attacks on inner rounds of DES [EB/OL]. (2010-01) . https://www. researchgate. net/publication/264885998_Fault-Algebraic_Attacks_on_Inner_Rounds_of_DES.

[3] 黄静, 赵新杰, 张帆, 等. PRESENT 代数故障攻击的改进与评估 [J]. 通信学报, 2016, 37 (8): 144-156. (Huang Jing, Zhao Xinjie, Zhang Fan, *et al.* Improvement and evaluation for algebraic fault attacks on PRESENT [J]. Journal on Communications, 2016, 37 (8): 144-156.)

[4] Zhao Xinjie, Guo Shize, Zhang Fan, *et al.* Algebraic fault analysis on GOST for key recovery and reverse engineering [C]// Fault Diagnosis and Tolerance in Cryptography. 2014: 29-39.

[5] 冀可可, 王韬, 赵新杰, 等. 轻型分组密码 LED 代数故障攻击方法 [J]. 计算机应用研究, 2013, 30 (4): 1183-1186. (Ji Keke, Wang Tao, Zhao Xinjie, *et al.* Algebraic fault attack on LED light-weight block cipher [J]. Application Research of Computers, 2013, 30 (4): 1183-1186.)

[6] Yang Gangqiang, Zhu Bo, Suder V, *et al.* The simeck family of lightweight block ciphers [C]// Proc of International Workshop on Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2015: 307-329.

[7] Biryukov A, Perrin L P. State of the art in lightweight symmetric cryptography [EB/OL]. [2018-01-12]. https://eprint. iacr. org/2017/511. pdf.

[8] Bagheri N. Linear cryptanalysis of reduced-round SIMECK variants [M]// Progress in Cryptology. [S. l.] : Springer International Publishing, 2015: 140-152.

[9] Zhang Kai, Guan Jie, Hu Bin, *et al.* Security evaluation on simeck against zero-correlation linear cryptanalysis [J/OL]. [2018-01-12]. https://eprint. iacr. org/2015/911.

[10] 陈彦琴, 张文英. SIMECK32//64 算法的不可能差分分析 [J]. 计算机工程, 2017, 43 (4): 141-144. (Chen Yanqin, Zhang Wenying. Impossible differential cryptanalysis of SIMECK32//64 algorithm [J]. Computer Engineering, 2017, 43 (4): 141-144.)

[11] Qiao Kexin, Hu Lei, Sun Siwei. Differential security evaluation of simeck with dynamic key-guessing techniques [C]// Proc of International Conference on Information Systems Security and Privacy. 2015: 74-84.

[12] Nalla V, Sahu R A, Saraswat V. Differential fault attack on SIMECK [C]// Proc of Workshop on Cryptography and Security in Computing Systems. New York: ACM Press, 2016: 45-48.

[13] Sadhukhan R, Patranabis S, Ghoshal A, *et al.* An evaluation of lightweight block ciphers for resource-constrained applications: area, performance, and security [J]. Journal of Hardware & Systems Security, 2017 (4): 1-16.

[14] 郭世泽, 王韬, 赵新杰. 密码旁路分析原理与方法 [M]. 北京: 科学出版社, 2014. (Guo Shize, Wang Tao, Zhao Xinjie. Principles and methodologies of side-channel analysis in cryptography [M]. Beijing: Science Press, 2014.)

[15] Joye M, Tunstall M. Fault analysis in cryptography [M]. Berlin: Springer, 2012.